On Hardware Evolvability and Levels of Granularity

Adrian Stoics
Center for Space Microelectronics Technology
Jet Propulsion Laboratory,
California Institute of Technology
Pasadena, CA, 91109

ABSTRACT
Evolvable hardware addresses hardware that self-organizes/ reconfigures under the guidance of evolutionary mechanisms. Some experiments in evolving at transistor level are briefly presented and the perspective of transistors as functional approximators is suggested. In a broader context one analyses approaches to evolvable hardware specifying the level of granularity at which evolution will operate, in accordance with the level of design abstractions in the modeling hierarchy: primitive, functional and behavioral levels in simulated circuits, and transistor, subcircuit and high level function in hardware. Comments are made linking the role of Automatically Defined Functions in hardware evolution, the process of achieving higher/coarser levels of granularity in the semiotic perspective, and the evolution of modularity of organismic designs. Finally, it is suggested testing the effect of changing levels of granularity during hardware evolution,

KEYWORDS: *evolvable hardware, granularity, evolvability*

1. EVOLVABLE HARDWARE

Evolution appears to be nature's solution to design. Being able to replicate such a capability in an artificial system would offer tremendous insights into ourselves and a powerful tool for building adaptive, intelligent systems.

From the perspective of space exploration, empowering spacecraft with adaptive, intelligent capabilities is invaluable for autonomy. Adaptive features are needed to cope with the uncertainty of spacecraft remote operating conditions, performing totally unexpected functions, and for fault-tolerance. The on-board computer needs to be able to solve problems for which solutions were not specified on ground, and command the spacecraft to adapt to new situations. For adaptive, versatile spacecraft, electronic hardware must posses the capability to reconfigure, or moreover, to self-reconfigure, as needed,

Evolvable hardware (EHW) is adaptive hardware that reconfigures under the control of an evolutionary algorithm [1]. *Extrinsic* EHW refers to evolution in a software simulation using models of the hardware behavior, downloading the configuration of the best evolved architecture to programmable hardware. in *intrinsic* hardware (to which most of the following discussion applies) configuration bits are iteratively downloaded to hardware, evaluating a degree of adaptation/fitness by observing the behavior of the real hardware.

Hardware evolution is performed through a succession of changes of elementary cell functions and cell inter-connectivity pattern, thus obtaining increasingly more fit configurations until a target functionality is reached. As it is the case in nature, evolution results in individuals that are increasingly more adapted to their environments, and can change themselves to match changes in environments and modifications of their own goals. Unlike in nature, evolution in silicon has the advantage that could be extremely rapid, with millions of generations of "living" circuits evaluated in only a few seconds.

Hardware evolution can be seen as an on-chip search for the circuit/configuration whose behavior is closest to the required one (e.g. gives best performance/adaptation to the environment). The suitability for a parallel hardware implementation of evolvable hardware, with multiple "islands" of concurrently evolving circuits on the same chip, or in a multi-chip or stacked configuration is very attractive.

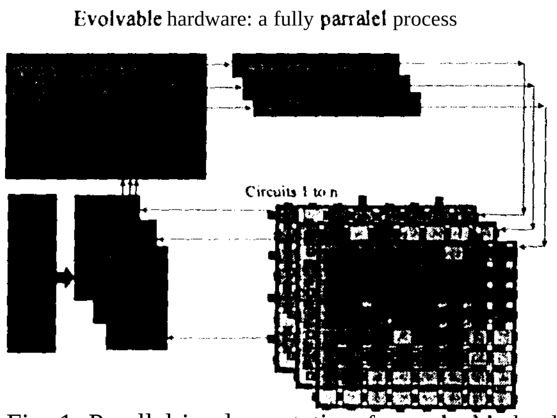

Fig. 1. Parallel implementation for evolvable hardware

The granularity of hardware building blocks for those attempting intrinsic evolvable hardware is currently influenced by the availability of certain programmable devices. The paper presents results of simulated evolution at transistor level and discusses on the role of the level of granularity and evolvability.

2. EXPERIMENTS IN EVOLVING CMOS CIRCUITS AT TRANSISTOR LEVEL

Successful evolution has been reported in simulations (analog [2], [3] and digital [4]) and in real hardware [5] [6]. In the intrinsic EHW perspective, the focus has been on using commercially available FPGAs (Field Programmable Gate Arrays) but custom designs of chips using higher-level functional blocks are also reported [6]. A collection of papers dedicated to the subject is [18].

In here the distinct focus of attention is on issues related to designing the reconfigurable part of an evolvable CMOS chip. The choice is motivated by the fact that CMOS technology is the basis of today's microelectronics industry, and NMOS/PMOS transistors are the elementary components of both analog and digital designs.

One can look at MOS transistors from the perspective of function approximators. Function approximation has recently been approached with computational intelligence techniques, demonstrating general approximation capabilities for structures of neural networks [7] [8] and fuzzy systems [9] [10]. As hardware implementations ultimately rely on silicon, a general functional approximator (FA) implemented in hardware will ultimately be relying on (e.g.) MOS transistors.

A set of experiments was performed to investigate evolution-related issues at transistor and simple sub-circuit level. The objective was to evolve a circuit that provided a bell-shaped response when the input increased linearly. This response can be obtained for example with the circuit in Fig. 2, with one input kept at constant voltage and the other increasing linearly. The evolution was performed on simulated circuits (using SPICE). Constraints were imposed on the mechanism generating the circuits such that all the circuits produced were SPICE simulatable (this differs than Koza's experiments where non-simulatable circuits are eliminated by evolution (e..g in [1 1]). A limitation existed also on the number of circuits evaluated, which was much smaller than those reported by Koza (-104 compared to -1 $0^{7)}$. .

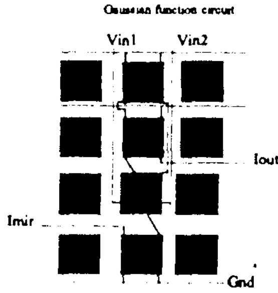

Fig. 2 A circuit producing a bell-shaped response

In the first set of experiments the circuit topology was considered known, and evolution concerned two types of parameters: transistor channel Width and Length. These parameter domains are discrete (they are a multiple of the feature size), and for this case a total of 8 possible widths and 8 lengths was considered. This led to a (3+3)*8=48 bit coding for the circuit. Applying a Genetic Algorithm on a population of 50 individuals, converged easily to good solutions (an illustration of the population after about 200 generations is given in Fig. 3. This is explained by the fact that, although the region searched is small (~2^{10} in the search space of 2^{48} possible combinations), marry combinations are good; in this case the topology is the fundamental factor in circuit behavior.

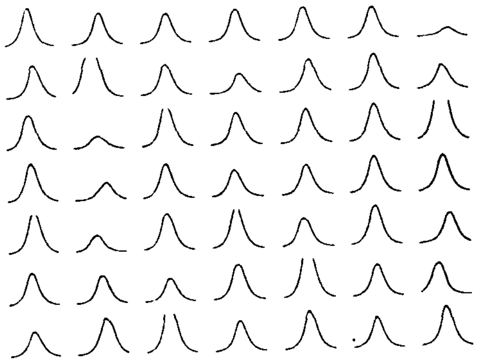

Fig. 3 Response of a population of 49 circuits after 200 generations

The problem becomes complicated when one tries to evolve the topology. Several alternatives were tried, briefly described in the following:

- A 2D transistor array was considered, which resembled an existing FPGA model (Xilinx 62 16). The code for each cell specified the type of the cell which could be a transistor with a certain orientation, or a type of wire routing (these are illustrated in Fig. 4). A 2D chromosome was associated to the array. Only evolution by GA was attempted. The problem appears in specifying 2D crossover, determining which 2D zones should be swapped. Crossover must consider the fact that the new circuits must match their connections at cell borders.

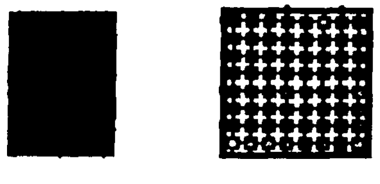

Fig. 4 A transistor array patterned after an FPGA

- A leveled architecture (with a matrix arrangement as in Fig. 2) where components on a given level were from a list of allowed components (see Fig, 5). Components were low-level subcircuits (current mirror, differential pair, pair of transistors). The coding was such that the possible connections were limited to those that made sense (e.g. the Source terminal of transistor in level 4 could not connect to VDD).

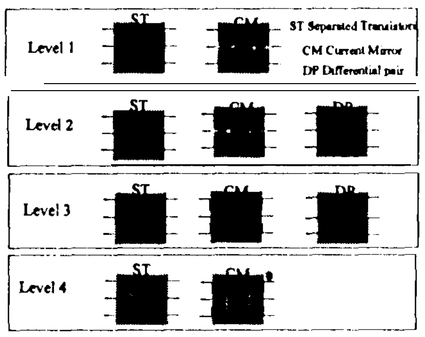

Fig. 5 Components allowed at different levels

The experiments can be interpreted as follows:

- When the topology of the electronic circuit is given, the transistor parameters (in this case PMOS/NMOS channel Width and Length) can be easily obtained through evolution.
- If one attempts evolution in a space restricted at the size of the human-designed solution (in our case when the circuits were limited at 8 transistors) it was not possible to evolve a circuit from specifications. The methods attempted included genetic algorithms, and a search based on orthogonal arrays. With the representation used, the search space appears very much as a flat region, with a singular spike for the solution circuit, Its neighbors, with genetic code differing even one bit from the solution had extremely low fitness; i.e. changing even one connection between 2 transistors had a dramatic effect in circuit behavior. This is possibly a consequence of the search around the optimal size (minimal number of transistors) solution. If the circuit would have contained redundant circuitry changes in those regions would have produce less significant effects.

It was however possible to evolve the topology using genetic programming and an embrionic, growth-based approach, in which the number of components (size of the circuit) was not restricted (results obtained by Koza's group, not published yet). The first set of simulations led to a circuit with 36 transistors.

Evolving on a structure with fixed, minimal number of transistors for the function (for which a solution circuit is already known) suffers from the fact that any mutation involving a connection/topology change leads to a dramatically different response. One possible way in which this can be alleviated is to allow gradual connections (e.g. connections modeled by resistors in [0 100G]) during the evolution. Thus may be architecture which allows gradual transitions between topologies are possible. It could look like a densely-connected "sea of transistors" in which components are connected by a "fuzzy" wire (i.e. with values in [0,1] rather than {0,1 }), appearing much like a neural network, but with transistor instead of neurons. The graded connection would have a catalyst role during evolution; it smoothens the search space around the solutions (in this respect neural architectures using gradual connections/weights appear suitable for evolution).

3. HARDWARE LEVELS OF GRANULARITY AND EVOLUTION

A fundamental question is how does the choice of a level of design abstraction influence hardware "evolvability"?

(The search/optimization algorithm may be not that important: the "no free lunch" theorems establish that for any algorithm, any elevated performance over one class of problems is offset by performance over another class [12].) When evolution takes place directly in the component space the choice of the primitive building blocks affects the evolvability (e.g. evolving a NN appears easier than evolving a circuit with transistors).

The choices for the level at which evolution process could operate are as follows.

For evolving simulated circuits. The levels of design abstraction in the modeling hierarchy are [13]:

- Primitive Devices (Transistor level - MOS,BJT,etc) represented by analytical equations or tables
- Functional Macromodels (e.g. Op. Amp. Level) derived by circuit simplification, circuit build-up, symbolic methods
- Behavioral High level language descriptions - linear and nonlinear mathematical equations, tables, etc.

Evolution can be made at a certain level and then the design converted for hardware implementation with appropriate compilers and synthesis tools.

For evolving directly in hardware. A similar selection of level of primitives is available in a reconfigurable structure (specifying what should be the "building blocks"):

- Transistor level. Some simulations were discussed in the previous section.
- Subcircuit level. This is the Op. Amp./digital gates level. Succesful hardware evolution taking place on an FPGA chip was reported by Thompson [5]. Versatile FPAA (Field Programmable Analog Arrays) are still missing, but are expected to appear on the market within 2-3 years.

. High level/ functional level One way of coping with difficulties in evolving circuits is to provide high level building blocks, filters, modulators in the analog domain, or adders, MUXS in the digital ones. This approach is followed by Higuchi [6].

The idea of modularity, and changing granularity during evolution may be key to hardware to evolvability (in intelligent systems, knowing presumes changing resolution or granularity [14]). In the biological world modularity is most likely the result of evolutionary modifications [15]. In the evolvable hardware context, Koza has indicated the usefulness of Automatically Defined Function (ADF) for evolving analog electronic circuits (an ADF is "a function (subroutine, module, etc.) that is dynamically evolved during a run of genetic programming and that maybe called by a program that is concurrently being evolved") [16]. In effect the idea that no approach to automated programming (and consequently hardware evolution) is likely to be successful on non-trivial problems unless it provides "some hierarchical mechanism to exploit, by reuse and parametrization, the regularities, symmetries, homogeneities, similarities, patterns, and modularities inherent in problem environments" is central to Koza's book [17].

When choosing representation levels, it may be interesting to consider having simultaneous representations at different levels of granularity. During evolution one may switch between levels of granularity depending which one offers advantages at that time.

4. CONCLUSION

The paper presented simulation results and discussed issues of evolution at transistor level. While parametric evolution appears easy, topological evolution is hard. Different levels of granularity are available for evolvable hardware. It was suggested that changing/switching levels of granularity during evolution may help.

5. ACKNOWLEDGEMENT

The research described in this paper was performed by the Center for Space Microelectronics Technology, Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the JPL Director's Discretionary Fund.

6. REFERENCES

[1] De Garis, H. "Evolvable Hardware: Genetic Programming of a Darwin Machine". In *Proc. Of the Int. Conf. On Artificial Neural Networks and Genetic Algorithms, Innsbruck, Austria,* Springer Verlag, 1993

[2] Grimbley, J. B. Automatic Analogue Network Synthesis using Genetic Algorithms, 1st IEE/IEEE Conf: Genetic Algorithms in Engineering Systems, UK, 1995

[3] Koza, J., Bennett Ill, F. H., Lohn J., Dunlap, F., Keane M. A., and Andre, D. "Automated Synthesis of Computational Circuits Using Genetic Prograsnming". In ***Prac of Second Annual Genetic Programming Conference,*** Stanford July 13-16, 1997 (to appear)

[4] Hemmi, H., Hikage, T. and Shimohara, K. AdAM: A Hardware Evolutionary System , In *Proc. of ICEC,* (193-196), 1997

[5] Thompson, A., Silicon Evolution. In ***Proc. of First Annual Genetic Programming Conference,*** Stanford, 1996

[6] Higuchi, T., Murakawa, M., Iwata, M., Kajitani, I., Liu, W. and Salami, M., "Evolvable Hardware at Function Level." *In Proc. of ICEC,* (1 87-192), 1997

[7] Cybenko, G. Approximation by Superpositions of a Sigmoidal Function. ***Mathematics of Control. Signals, and Systems, 2 (303-3 14), 1989***

[8] Homik, K., Stinchombe, M., and White, H Multilayer Feedforward Networks are Universal Approximators Neural Networks, 2 (359-366), 1995

[9] Kosko, B. Fuzzy Systems as Universal Approximators. In ***Proc. of the 1st IEEE Conference on Fuzzy Systems,(1153-1162),*** 1992

[10] Wang, L. X. Fuzzy Systems are Universal Approximators. In ***Proc. of the 1st IEEE Conference on Fuzzy Systems, (11*** 63-1 170), 1992

[11] Koza, J., Bennett 111, F. H., Andre, D. and Keane, M., Automated WYWIWYG Design of Both the Topology and Component values of Electrical Circuits Using Genetic Programming, In ***Proc. of First Annual Genetic Programming Conference,*** Stanford, (123- 131), 1996

[12] Wolpert, D. H. and Macready, W. G., "No Free Lunch Theorems for Optimization", ***IEEE Transactions on Evolutionary Computation,*** Vol. 1. No. 1, (67-82), 1997

[13] Mantooth, H. A. and Fiegenbaum, M., ***Modeling with an Analog Hardware Description Language.*** Kluwer Acedemic Publishers, 1995

[14] Meystel, A. "What is 'semiotics' after all? (Learning how to know: semiotics and multiscale cybernetics)". Semiotcs97 Internet at http: //isd.cme.nist gov/ proj/semiotics/isas97.html

[15] Wagner, G. P. and Altenberg, L., "Perspective: Complex Adaptation and the Evolution of Evolvability". In ***Evolution, 50(3), (967-976), 1996***

[16] Koza, J., Andre, D., Bennett III, F. H., and Keane, M. A., Use of Automatically Defined Functions and Architecture-Altering Operations in Automated Circuit Synthesis with Genetic Programming. In ***Proc. of First Annual Genetic Programming Conference,*** Strmford, (132- 141), 1996

[17] Koza, J. R Genetic ***Programming 11: Automatic Discovery of Reusable Programs.*** Cambridge, MA: MIT Press, 1994

[18] Sanchez, E. and Tomrrssini, M. (Eds), ***Towards Evolvable Hardware - The Evolutionary Engineering Approach.*** Springer, 1996,